18/09

SOURCES OF APPROXIMATION

1) MEASURE ERRORS ⟶ measure instrument
2) ARITHMETIC ERRORS ⟶ propagation of rounding errors operation by operation
3) TRUNCATION ERRORS ⟶ truncation of infinite
4) INHERENT ERRORS ⟶ finite representation of data

(ABSOLUTE VS RELATIVE) ERROR

ABSOLUTE : $E_x = \tilde{x} - x$

RELATIVE : $R_x = \frac{\tilde{x} - x}{x} \quad (x \neq 0)$

where $\tilde{x} \simeq x \longrightarrow \tilde{x} = x(1 + R_x)$

ACCURACY VS PRECISION , (TRUNCATION VS ROUNDING) ERROR NO!!!

REPRESENTATION OF REAL-NUMBER IN BASE β

HP: $\beta > 1$, $x \neq 0$, $0 \leq d_i \leq \beta - 1$, $d_1 \neq 0$ and not all $d_i = \beta - 1$

if not, you may have infinite representations of the same number by adding zeros

$0.d_1 d_2 d_3 \ldots \longleftarrow \frac{1}{\beta} \leq m < 1 \longleftarrow$ mantissa ; exponential part

TH: $x = sign(x)(d_1\beta^{-1} + d_2\beta^{-2} + \ldots)\beta^p = sign(x)m\beta^p$ ⟶ UNIQUE REPRESENTATION:

FLOATING-POINT NUMBER

$\mathcal{F}(\beta, t, L, U)$ system of floating points

- base
- precision
- exponent range

The floating point $x = \pm(d_1\beta^{-1} + \cdots + d_t\beta^{-t})\beta^p, \quad L \leq p \leq U, \quad 0 \leq d_i \leq \beta - 1, \quad i = 1, \ldots, t$

$(d_1 \ldots d_t) = m$ ⟶ mantissa

exponent of characteristics

It is normalized when $d_1 \neq 0$ ⟶ $\frac{1}{\beta} \leq d_1 < 1$ ⟶ ???

REASONS

- Representation of each number unique.
- No digits wasted on leading zeros.
- Leading bit need not be stored (in binary system).

es: $x = 12.\overline{3}$

$\mathcal{F}(10, 5, -3, 3) = \pm(0.12333) \cdot 10^2$

TRUNCATED ← $3 < 5$

$x = 12.\overline{8}$

$\mathcal{F}(10, 5, -3, 3) = \pm(0.12889) \cdot 10^2$

$8 > 5$ → ROUNDED

PROPERTIES OF $\mathcal{F}(\beta,t,L,U)$

1) System is finite and discrete
2) TOT_NUM_PTS = $2(\beta-1)\beta^{t-1}(U-L+1)+1$
3) SMALLEST_NUM_POSITIVE = $\text{UFL} = \beta^{L-1}$ $\longrightarrow$ $\mathcal{F}(\beta,t,L,U)$ << $\longrightarrow$ m << and t << (m: $0.00\ldots$ with $1/\beta$; t: L)
   ↳ $\text{UFL} = \pm m \cdot \beta^t = + \beta^{-1} \cdot \beta^L = \beta^{L-1}$
4) LARGEST_NUM = $\text{OFL} = \beta^U(1-\beta^{-t})$ $\longrightarrow$ $\mathcal{F}(\beta,t,L,U)$ >> $\longrightarrow$ m >> and t >> (m: $0.11\ldots$; t: U)
   ↳ $\text{OFL} = \pm m \cdot \beta^t =$ ....
5) Numbers are equally spaced only between successive powers of β
6) Not all real numbers are exactly representable $\longrightarrow$ only MACHINE NUMBERS are
   ↳ elements of $\mathcal{F}(\beta,t,L,U)$

ROUNDING RULES

1) CHOP or TRUNCATION
2) ROUND TO NEAREST

STANDARD IEEE

1) SINGLE PRECISION $\longrightarrow$ 4 byte = 32 bit = 1 (sign) + 23 (m) + 8 (t) $\quad F(2, 24, -128, 127)$
2) DOUBLE PRECISION $\longrightarrow$ 8 byte = 64 bit = 1 (sign) + 52 (m) + 11 (t) $\quad F(2, 53, -1024, 1023)$

PS: in base 2, if it is normalized ($d_1 \neq 0$) d_1 must be 1, so it's not stored to have more space $\longrightarrow$ so mantissas are 23 (not 24) and 52 (not 53)

MACHINE PRECISION (ϵ_{mach})

If we represent numbers by:

1) CHOPPING $\longrightarrow$ $\epsilon_{mach} = \beta^{1-t}$
2) ROUNDING TO NEAREST $\longrightarrow$ $\epsilon_{mach} = \frac{1}{2}\beta^{1-t}$

DEF: ϵ_{mach} is smallest value of ϵ such that $fl(1+\epsilon) > 1$ and we could also write it as $\left|\frac{fl(x)-x}{x}\right| \leq \epsilon_{mach}$ (left side: $|R_x|$)

EXCEPTIONAL VALUES

In IEEE we have 'InF' and 'NaN'
('InF': $1/0$; 'NaN': $0/0$; $0 \cdot \text{Inf}$; Inf/Inf)

FLOATING POINT ARITMETIC

$x \odot y = fl(x \cdot y)$ where $\odot : F \times F \longrightarrow F$

which error is $\left| \frac{(x \odot y) - (x \cdot y)}{x \cdot y} \right| <$ machine precision of the finite set of numbers we are using

STEPS TO PERFORM $x \odot y$

where $\{x \in F, y \in F, \odot \in \{+, -, *, /\}\}$

1) $x \odot y$ in a register, which is a portion of memory which is double or larger than the portion of memory supposed to store x and y. Because when you perform operations you can have extra digits so you need more space to store them.
2) $fl(x \odot y) \in F$ we must compute the float of the result and this is stored in the finite set, getting truncated or rounded $\longrightarrow$ error

CANCELLATION

It's the subtraction between 2 t-digits numbers with same sign and similar magnitude. It returns a result with less than t-digits $\longrightarrow$ EXACTLY REPRESENTABLE BUT WITH SERIOUS LOSS OF INFORMATION

es $x = 1.92403 \cdot 10^2$, $y = 1.92275 \cdot 10^2$

$fl(x) = 0.192403 \cdot 10^3$, $fl(y) = 0.192275 \cdot 10^3$

Then $z = fl(x-y) = fl([1.92403 - 1.922753] \cdot 10^2) = fl(0.00128 \cdot 10^2) = 0.128$

$fl(z) = 0.128000$

19/09

LINEAR SYSTEMS

It's written as $Ax = b$ also written as $\underset{m \times n}{\begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix}} \cdot \underset{n \times 1}{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}} = \underset{m \times 1}{\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}}$ $m \geq n$

$$\begin{cases} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n = b_1 \\ \vdots \\ A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n = b_m \end{cases}$$

We'll see:

1) EXISTANCE / UNIQUENESS OF SOLUTION
2) NUMERICAL METHODS FOR SOLUTION
3) CONDITIONING OF THE PROBLEM

SQUARE LINEAR SYSTEMS

$\det(A) \neq 0$ and A invertible — rank is maximum

HP: $m = n$, (A not singular | $\operatorname{rank}(A) = n$ | $Ax = 0$ only for $x = 0$)

TH: $b = (b_1, \dots, b_n)$ exists and is unique

RESULT COMPUTATION

1) $Ax = b \longrightarrow A^{-1}Ax = A^{-1}b \longrightarrow x = A^{-1}b$ (COFACTOR MATRIX; EXPENSIVE IF n>>)
2) $x_i = \dfrac{\det(A_i)}{\det(A)}$ (ALSO)

- So we'll consider 2 methods.
  1) DIRECT ⟶ FINITE STEPS × SOLUTION, MORE PRECISE, EXPENSIVE COST
  2) ITERATIVE ⟶ INFINITE STEPS × SOLUTION, LESS PRECISE, LESS EXPENSIVE

- In order to perform error analysis:
  1) ROUNDING/ARITHMETIC ERRORS $\propto$ NUM_ALGORITHM_STEPS (IT'S CALLED STABLE IF WE COULD SAY THAT TOTAL_ERROR < ε)
  2) INHERENT ERRORS ~~$\propto$~~ NUM_ALGORITHM_STEPS (IT'S CALLED ILL-POSED PROBLEM IF ERROR >>)

DIRECT METHODS

From $Ax = b$ let $A = LU$ so that we have $\begin{cases} Ly = b \\ Ux = y \end{cases}$ (Ly = b: FORWARD SUBSTITUTION ALGORITHM; Ux = y: BACKWARD SUBSTITUTION ALGORITHM)

$LUx = b$ $\quad O\left(\frac{n^3}{3}\right)$

THE TOTAL COST IS $O\left(\frac{n^3}{3}\right) + 2 \cdot O(n^2)$ (DOMINANT)

Where L is lower triangular matrix → FORWARD RESOLUTION

Where U is upper triangular matrix → BACKWARD RESOLUTION

$O(n^2)$

When we have pivoting algorithm ⟶ $PA = LU$ ⟶ $PAx = Pb$ ⟶ $\begin{cases} Ly = Pb \\ Ux = y \end{cases}$

(P: PERMUTATION MATRIX, identity matrix with some rows swapped)

So we have 3 steps:

1) $PA = LU$
2) $Ly = Pb$
3) $Ux = y$

ITERATIVE METHODS

The idea is to build a vector using convergence $x^* = \lim_{k \to \infty} x_k$ (EXACT SOLUTION; $g(x_{k-1})$ IN GENERAL)

We have different examples depending on their $g(x_{k-1})$:

1) STATIONARY ITERATIVE METHODS

$x_{k+1} = B x_k + f$ (B: ITERATION MATRIX; f: VECTOR FROM b)

$O(n^2) * p$ (p: NUMBER OF PERFORMED ITERATIONS) ⟶ THE SMALLER IS p, THE QUICKER IS THE ALGORITHM

2) GRADIENT LIKE METHODS

SKIPPATA

We cannot iterate forever so we must know when to stop, in order to do so the are **STOPPING CRITERIA**:

1) $\|r_k\| \leq \varepsilon$
2) $\frac{\|r_k\|}{\|b\|} \leq \varepsilon$

(ε: TOLERANCE $\approx [10^{-5}, 10^{-10}]$; $r_k = b - Ax_k$)

3) $\|x_{k+1} - x_k\| \leq \tau$

ARRAY NORMS

$\|\cdot\| : \mathbb{R}^n \longrightarrow \mathbb{R}$

1) $\|x\| \geq 0$ and $\|x\| = 0 \leftrightarrow x = 0$

2) $\|x+y\| \leq \|x\| + \|y\|$

3) $\|\lambda x\| = |\lambda| \cdot \|x\|$

EUCLIDEAN NORM :

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

MANHATTAN NORM :

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

INFINITE NORM :

$$\|x\|_\infty = \max_{1 \leq i \leq n} \{|x_i|\}$$

MATRIX NORMS

$\|\cdot\| : \mathbb{R}^{m \times n} \longrightarrow \mathbb{R}$

1) $\|A\| \geq 0$ and $\|A\| = 0 \leftrightarrow A = \emptyset$

2) $\|A+B\| \leq \|A\| + \|B\|$

3) $\|\lambda \cdot A\| = |\lambda| \cdot \|A\|$

FORBENIUS NORM :

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$$

2-NORM:

NON È MOLTIPLICAZIONE MA INDICA DI A CHI SI RIFERISCE

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

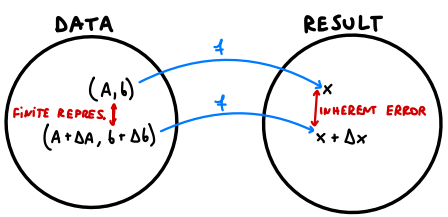
AUTOVALORE

THE LARGEST EIGEN VALUES IN ABSOLUTE VALUE

1-NORM :

$$\|A\|_1 = \max_{1 \leq j \leq n} \left\{ \sum_{i=1}^{m} |a_{ij}| \right\}$$

INHERENT ERROR



When the error in the result is much grater than the error in the data we have a problem, otherwise it's normal.

So the problem is when $\|\Delta x\| \gg \|\Delta A\| + \|\Delta b\|$ → PROBLEM IS ILL-CONDITIONED

RESULT :

$$\left\| \frac{\Delta x}{x} \right\| \leq \|A^{-1}\| \cdot \|A\| \cdot \frac{\|\Delta b\|}{\|b\|} = K(A) \frac{\|\Delta b\|}{\|b\|}$$

CONDITION NUMBER OF A MATRIX

RELATIVE MEASURE OF THE ERROR

RELATIVE MEASURE OF THE DATA

$K(A)$ small → WELL-CONDITIONED

$K(A)$ large → ILL-CONDITIONED

21/09

DOT PRODUCT

$f : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$

$$x^\top y = \sum_{i=1}^{n} x_i y_i$$

SYMMETRIC, POSITIVE DEFINITE MATRICES

It's a matrix $A \in \mathbb{R}^{n \times n}$ where:

1) $A = A^T \longrightarrow$ SYMMETRIC
2) $\forall x \in V - \{\emptyset\} : x^T A x > \emptyset \longrightarrow$ DEFINITE POSITIVE (POSITIVE SEMI DEFINITIVE: $\geq \emptyset$)

And it follows those properties:

1) $\text{Ker}(A) = \{\emptyset\}$ where $\text{Ker}(A) := \{x \in \mathbb{R}^n : Ax = \emptyset\}$
   BECAUSE $\longrightarrow x^T A x > \emptyset \ \forall x \neq \emptyset \longrightarrow Ax \neq \emptyset$ if $x \neq \emptyset$
2) $a_{ii} > \emptyset$
   BECAUSE $\longrightarrow a_{ii} = e_i^T A e_i > \emptyset$ (e_i = i-th vector of standard basis in $\mathbb{R}^n$)
3) If the eigenvalues of $A \in \mathbb{R}^{n \times n}$ symmetric are all positive, then A is positive definite

ANGLES AND ORTOGONALITY

The scalar product can provide us information about the angles between 2 vectors: $\cos\omega = \dfrac{\langle x, y \rangle}{\|x\| \, \|y\|}$.

If $\omega < \pi$ then x, y have the same orientation and vice-versa.

ORTOGONALITY: x, y are ortogonal $\longleftrightarrow \langle x, y \rangle = \emptyset$

ORTONORMALITY: x, y are ortogonal and $\|x\| = 1 = \|y\|$

ORTOGONAL MATRIX

It's a matrix $A \in \mathbb{R}^{n \times n}$ where its column are ortonormal $\longrightarrow A A^T = I = A^T A \longrightarrow A^{-1} = A^T$

1) $\|Ax\|^2 = \|x\|^2 \longrightarrow$ the lenght of a vector x is not changed when transforming it by orthogonal matrix.
2) $\cos(\omega) = \dfrac{x^T y}{\|x\| \, \|y\|} \longrightarrow$ the angle between Ax and Ay is the angle between x and y

ORTHONORMAL BASIS

$C_1v_1 + \dots + C_nv_n = 0 \longleftrightarrow C_1 = \dots = C_n = 0$ (linear independent)

Giving a vector space "V" of dimension n, $\{v_1, \dots, v_n\}$ is a basis of V

DEF: $\{b_1, \dots, b_n\}$ is a basis of V if $\begin{cases} \langle b_i, b_j \rangle = 0 & \text{for } i \neq j \\ \langle b_i, b_i \rangle = 1 \end{cases}$

ORTHOGONAL COMPLEMENT

If V vector space and $\mathcal{U}$ vector space: $\mathcal{U} \subset V$ (dim = D, dim = M) then $\mathcal{U}$ is a subspace.

Then there is $\mathcal{U}^\perp$ which is a V subspace which contains all vector in V that are orthogonal to every vector in $\mathcal{U}$ and $\mathcal{U} \cap \mathcal{U}^\perp = \{0\}$

ORTHOGONAL PROJECTIONS

They are important in machine learning because of DATA REDUCTION.

We have to deal with high-dimensional data which is hard to analyze. → WE NEED TO CHANGE THE DATA → COMPRESSION

- We project the original high-dim data in low-dim space to learn more about dataset and extract relevant patterns

DEF: $U \subseteq V$, a linear mapping $\pi: V \longrightarrow U$ it's an ORTOGONAL PROJECTION if $\pi \circ \pi = \pi$ (composed)

↳ can be expressed by transformation matrices → PROJECTION MATRICES → $P_\pi^2 = P_\pi$

PROJECTION ONTO GENERAL SUBSPACES

Let $V = \mathbb{R}^n$, so we look at vectors $x \in \mathbb{R}^n \geq U$, where $\dim(U) = m \geq 1$

Assume $(b_1, \dots, b_m)$ an ordered basis of U, so $\pi_U(x)$ is any projection of x onto U and so it should be necessarily be an element of U.

↳ $\pi_U(x) = \sum_{i=1}^{m} \lambda_i b_i$ → LINEAR COMBINATION OF $(b_1, \dots, b_m)$

EIGENVALUES AND EIGENVECTORS

Let $A \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is EIGENVALUES of A and $x \in \mathbb{R} - \{0\}$ is the EIGENVECTOR if $Ax = \lambda x$

We have 4 equivalent statements

1) λ is EIGENVALUE of $A \in \mathbb{R}^{n \times n}$
2) $\exists\, x \in \mathbb{R}^n - \{0\} : Ax = \lambda x$
3) $(A - \lambda I_n)x = 0$ with $x \neq 0$
4) $rk(A - \lambda I_n) < n$
5) $\det(A - \lambda I_n) = 0$

THE EIGENVECTORS $(x_1, \dots, x_n)$ OF $A \in \mathbb{R}^{n \times n}$ WITH n DIFFERENT $(\lambda_1, \dots, \lambda_n)$ ARE LINEARLY INDEPENDENT

- SYMMETRIC, POSITIVE DEFINITE MATRICES → POSITIVE, REAL EIGENVALUES
- If x is an eigenvector of A associated with eigenvalue λ, also cx will be an eigenvector of A with the same eigenvalue

EIGENSPACE AND EIGENSPECTRUM

EIGENSPACE: the subspace of $\mathbb{R}^n$ spanned by the set of eigenvectors of A with eigenvalues λ

EIGENSPECTRUM: the set of all eigenvalues of A

GEOMETRIC MULTIPLICITY EIGENVALUE

It's the number of linearly independent eigenvectors associated with λ_i.

DEFECTIVE MATRIX

A square matrix $A \in \mathbb{R}^{n \times n}$ is defective if it possesses fewer than n linearly independent eigenvectors.

SPECTRAL THEOREM

HP: $A \in \mathbb{R}^{n \times n}$ symmetric

TH: $\exists$ an orthonormal basis of the corresponding vector space V consisting of eigenvectors of A and each eagenvalues is real.

IN OTHER WORDS: IF A is symmetric, then the matrix, containing its eigenvetors, is orthogonal and the eigenvalues are real

SIMILAR MATRICES

A, D are similar if exists an invertible one P so that $D = P^{-1}AP$

- A, D so will have the same eigenvalues

DIAGONALIZABLE MATRIX

$A \in \mathbb{R}^{n \times n}$ is diagonalizable if it's similar to a diagonal matrix $\longrightarrow$ $D = P^{-1}AP$

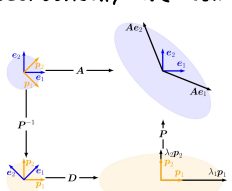
- Symmetnc matrix can always be diagonalized

EIGENDECOMPOSITION

$A \in \mathbb{R}^{n \times n}$ can be factored into $A = PDP^{-1}$ where D is a diagonal matrix with only eigenvalues of A on the diagonal

eigenvectors of A form a basis of $\mathbb{R}^n$

- Geometrically we can see matrix as a linear transformation between 2 vector spaces



25/09

SINGULAR VALUE COMPOSITION (SVD)

- More general theorem ⟶ applied to all matrices and it's always possible

HP: $A^{m \times n}$ and $\text{rank}(A) = r \in [0, \min\{m,n\}]$

TH:

$$\underset{m \times n}{A} = \underset{m \times m}{U} \; \underset{m \times n}{\Sigma} \; \underset{n \times n}{V^\top}$$

$V^\top$ → ORTHOGONAL MATRIX

U → ORTHOGONAL MATRIX

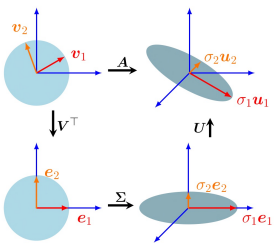
$\Sigma_{ii} = \sigma_i$, $\Sigma_{ij} = \emptyset$ for $i \neq j$ ⟶ CAN'T SPEAK PROPERLY ABOUT DIAGONAL MATRIX $(m \times n)$

BY CONVENTION ↓ $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0$

SINGULAR VALUES

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & \sigma_m & 0 & \dots & 0 \end{bmatrix}$$

$m - n$



PROPERTIES OF SVD

- There's no relation between eigenvalues and singular values of A, but there is between the singular values and the eigenvalues of A^TA

HP: $\lambda_i \geq 0$ eigenvalues of A^TA

TH: $A v_i = \sigma_i u_i$

DIM:

$(A \cdot B)^T = B^T A^T$

$$A^TA = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T U^T U \Sigma V^T = V\Sigma^T \Sigma V^T = V\Sigma^2 V^T \quad (U^TU = I)$$

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma V^T V \Sigma^T U^T = U\Sigma\Sigma^T U^T = U\Sigma^2 U^T \quad (V^TV = I)$$

Then we can obtain the SINGULAR VALUE EQUATION: $A = U\Sigma V^T \longrightarrow AV = U\Sigma V^T V$, $Av_i = \sigma_i u_i$

EIGENVALUE DECOMPOSITION VS SINGULAR VALUE DECOMPOSITION

EIGENVALUE DECOMPOSITION VS SINGULAR VALUE DECOMPOSITION

1) $\exists$ in $\mathbb{R}^{n \times n}$
2) Vectors in P are not necessarily orthogonal

1) $\exists$ in $\mathbb{R}^{m \times n}$
2) U and V are orthonormal

NOT NECESSARILY A ROTATION ← They are both compositions of three linear mappings ⟶ REPRESENTS A ROTATION

EIGENVALUES OF A ← $\lambda_i = \sigma_i^2$ → SINGULAR VALUES OF A

RELATION BETWEEN SINGULAR VALUES AND $\|A\|_2$

$$\|A\|_2 = \sqrt{\max\{|\lambda_1|,\dots,|\lambda_m|\}(A^TA)}$$

$\sigma_1, \dots, \sigma_n$ are sorted

$$\max\{\sigma_1,\dots,\sigma_n\}(A) = \sigma_1(A)$$

$$\|A^{-1}\|_2 = \sqrt{\rho((A^{-1})^T A^{-1})} = \sqrt{\rho((AA^T)^{-1})} = \frac{1}{\sigma_n(A)}$$

IF $\lambda_1, \dots, \lambda_n$ eigenvalues of A

THEN $\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_n}$ eigenvalues of A^{-1}

$$K_2(A) = \frac{\sigma_1(A)}{\sigma_n(A)}$$

MATRIX APPROXIMATION

We want to use SVD in a cheaper way than its own computation

HP: $A = U\Sigma V^T \in \mathbb{R}^{m\times n}$

TH: $\hat{A}(k) := \sum_{i=1}^{k} \sigma_i u_i v_i^\top = \sum_{i=1}^{k} \sigma_i A_i \simeq A$

PS: $\hat{A}(k)$ provides $k\cdot(1+m+n)$ numbers

DIM: $A_i = u_i \cdot v_i^T \quad (u_i \in \mathbb{R}^m, v_i \in \mathbb{R}^n)$

From HP:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T = \sum_{i=1}^{r} \sigma_i A_i \longrightarrow \hat{A}(k) = \sum_{i=1}^{K} \sigma_i A_i \simeq A \qquad K < r$$

28/09

SPECTRAL NORM OF A MATRIX

For $x \in \mathbb{R}^n - \{0\}$, the spectral norm of $A \in \mathbb{R}^{m \times n}$ is $\|A\|_2 := \max_x \frac{\|Ax\|_2}{\|x\|_2}$

DETERMINES HOW LONG ANY VECTOR x CAN BECOME WHEN MULTIPLIED BY A

SPECTRAL NORM AS APPROXIMATION

HP: $A \in \mathbb{R}^{m \times n}$ $\text{rank}(A) = r$, $B \in \mathbb{R}^{m \times n}$ $\text{rank}(B) = K$

TH: $\forall K : K \leq r$, $\hat{A}(k) = \sum_{i=1}^{K} \sigma_i u_i v_i^T$ we have that

1) $\widehat{A}(k) = \text{argmin}_{\text{rk}(B)=k} \|A - B\|_2$ — THE MINIMUM OVER THE SET OF ALL MATRICES OF $m \times n$ WITH RANK K REPRESENTING THE MATRIX WHICH HAS MINIMUM DISTANCE W RESPECT TO A INTO 2-NORM

2) $\left\|A - \widehat{A}(k)\right\|_2 = \sigma_{k+1}$

???

MACHINE LEARNING (LECTURE 4 - EVANGELISTA SITE)

- Machine Learning is the set of all tecniques and algorithms able to extract knowledge from the data and use it to make accurate decisions

The development of a Machine Learning algorithm follows some steps:

1 - UNDERSTANDING

A task is an unknown function $f(x)$. We need to approximate $y = f(x) \simeq f_\theta(x)$

1) is it learnable ? ⟶ if there's a real correlation between input (x) and output (y)
2) is it collectable ? ⟶ if we can collect data (x) efficiently and with scalability

2 - COLLECTION

Data requires time to be collected in good quality, usually it's necessary to clean up the dataset with some specific algorithm

We take data in .csv, we convert strings into numbers and then we collect all these numbers in a big matrix and we know how to manipulate them.

We have 2 way to encode STR ⟶ INT

1) INTEGER ENCODING ⟶ assign integer to value, summing doesn't make sense and creates undesired relations
2) ONE-HOT ENCODING ⟶ assign vector of integers to value, no unwanted relations and summing makes sense

After that the result is $X = [x^1 \; x^2 \; .. \; x^N] \in \mathbb{R}^{d \times N}$ (d - features, N - num of datapoints)

$y^i = f(x^i)$

$Y = [y^1 \; y^2 \; ... \; y^N]^T \in \mathbb{R}^N$

3 - DESIGN

1) SUPERVISED LEARNING ⟶ dataset composed by inputs (X) and outputs (Y) ⟶ CLUSTERING
2) UNSUPERVISED LEARNING ⟶ dataset composed by only inputs (X)

4 - TRAINING

We give data to let our model learn the patterns

5 - TUNING

It's the task of finding the good compromise over the flexibility of the model depending on the task.

OVERFITTING UNDERFITTING

6 - TESTING

$N_{DATA\_TOT} = N_{DATA\_TEST} + N_{DATA\_TRAIN}$ ⟶ SPLITTING

In order to maximize the accuracy of the prediction of your model :

MODEL FUNCTION — ACCURACY FUNCTION — MODEL OUTPUT — REAL OUTPUT

$$Acc(f_\theta) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \ell(f_\theta(x_i), k_{x_i})$$

28/09

LINEAR LEAST SQUARE PROBLEM (LSQ)

RESIDUAL ← $r \in \mathbb{R}^m$

If we have $Ax = b$ we can't compute $Ax - b = 0$ but we can find $\tilde{x}$: $A\tilde{x} - b \simeq 0$, so we have to find the smallest r possible $\longrightarrow$ $\min \{\|r\|_2\}$

So we have $\tilde{x} = \min_{x \in \mathbb{R}^n} \|r\|_2 = \min_{x \in \mathbb{R}^n} \|r\|_2^2$ → WE ELIMINATE THE SQUARE ROOT

! DON'T FORGOT THAT YOU ARE LOOKING FOR ARG()

HP : $A \in \mathbb{R}^{m \times n}$, $m > n$, $r = \text{rank}(A)$

TH : 1) $r = n$ $\longrightarrow$ unique solution

2) $r < n$ $\longrightarrow$ infinite solutions

DIM: 1) $r = n$

$n \times m$ symmetric positive definite matrix → IF $\text{rank}(A) = n$

$Ax = b \longrightarrow A^T A x = A^T b \longrightarrow$ UNIQUE SOLUTION $\longrightarrow$ we can use direct methods to compute it

2) $r < n$

$Ax^* = b \longrightarrow S = \{x \in \mathbb{R}^n : x \text{ solutions of } \min \{\|Ax - b\|_2^2\}\}$

$x^* = \min_{x \in S} \|x\|_2$ → WE USE SVD DECOMPOSITION

$x^* = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} \cdot v_i$

APPLICATION OF LSQ

y_i → OUTPUT MEASURED $\quad f(x_i)$ → OUTPUT FORECAST $\quad x_i$ → INPUT MEASURED / GIVEN

$f(x) = c_0 + c_1 x + c_2 x^2 + \dots + c_n x^n$

$\theta = \{c_0, c_1, \dots, c_n\}$

$r = \{r_0, r_1, \dots, r_n\} = \{y_0 - f(x_0), \dots, y_n - f(x_n)\} = \{y_0 - (c_0 + c_1 x_0 + \dots + c_n x_0^n), \dots, y_n - (c_0 + c_1 x_n + \dots + c_n x_n^n)\}$

So we have $\vec{r} = \vec{y} - A\vec{x}$

PARTIAL DIFFERENTATION

Given $f(x) = (f_1(x), \dots, f_n(x))$ then:

1) GRADIENT $\longrightarrow$ $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right)$ — CHAIN-RULE → $\frac{\partial (g \circ f)}{\partial x} = \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial x}$

2) HESSIAN MATRIX $\longrightarrow$

$$H_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

3) JACOBIAN MATRIX $\longrightarrow$

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

GRADIANT OF A MULTIVARIANT FUNCTION

LEAST-SQUARES FUNCTION

We consider a linear model $y = \Phi\theta \to \mathbb{R}^D$ (PARAMETER)

(y: OBSERVATIONS $\mathbb{R}^N$; Φ: $\mathbb{R}^{N \times D}$ INPUT FEATURES)

Then we consider $L(\theta) = \|y - \Phi\theta\|_2^2$ as LEAST-SQUARES FUNCTION

Now we look for $\partial L / \partial \theta \in \mathbb{R}^{1 \times D}$:

$$\frac{\partial L}{\partial \theta} = \underbrace{\frac{\partial L}{\partial (y - \Phi\theta)}}_{1} \cdot \underbrace{\frac{\partial (y - \Phi\theta)}{\partial \theta}}_{2} = 2(y - \Phi\theta)^T(-\Phi) = -2(y^T - \theta^T\Phi^T)\Phi$$

1) $$\frac{\partial \|y - \Phi\theta\|_2^2}{\partial (y - \Phi\theta)} = \frac{\partial \|e\|_2^2}{\partial e} = \left[\frac{\partial}{\partial e_1}\left(\sqrt{\sum_{i=1}^n e_i^2}\right)^2, \dots, \frac{\partial}{\partial e_n}\left(\sqrt{\sum_{i=1}^n e_i^2}\right)^2\right] = \left[\frac{\partial}{\partial e_1}\sum_{i=1}^n e_i^2, \dots, \frac{\partial}{\partial e_n}\sum_{i=1}^n e_i^2\right] =$$

$$= \left[\sum_{i=1}^n \frac{\partial e_i^2}{\partial e_1}, \dots, \sum_{i=1}^n \frac{\partial e_i^2}{\partial e_n}\right] = \left[2e_1, \dots, 2e_n\right] = 2e^T = 2(y - \Phi\theta)^T$$

2) $$\frac{\partial (y - \Phi\theta)}{\partial \theta} = -\Phi$$

02/10

BACKPROPAGATION AND AUTOMATIC DIFFERENTIATION

We can't use chain-rule explicitly for computing the gradient, because otherwise we would introduce unecessary hoverhead.

GRADIENT IN A DEEP NETWORK

The chain-rule is massively used in deep-learning, where we have much more function composed.

So we use **BACKPROPAGATION**

START

EXPAND IT WITH CHAIN-RULE

$$\frac{\partial L}{\partial \theta_{k-1}} = \frac{\partial L}{\partial f_k} \cdot \frac{\partial f_k}{\partial \theta_{k-1}} = \frac{\partial L}{\partial f_k} \cdot \frac{\partial f_k}{\partial \theta_{k-1}} \cdot \dots \cdot \frac{\partial f_{i+2}}{\partial f_{i+1}} \cdot \frac{\partial f_{i+1}}{\partial \theta_i}$$

SOLVE THE LAST ONE AND REUSE IT TO SOLVE THE PREVIOUS ONE UNTIL WE SOLVED THE 1°

AUTOMATIC DIFFERENTIATION

Backpropagation is a generalization of **AUTOMATIC DIFFERENTIATION**, which is a set of techniques to evaluate the gradient of function.

We use intermediate variables to handle the backpropagation of the chain-rule and we use the associativity of a matrix multiplication to handle the 2 directions of backpropagation

$$\frac{dy}{dx} = \frac{dy}{db} \cdot \left(\frac{db}{da} \cdot \frac{da}{dx} \right)$$

FORWARD MODE ⟶ gradients are expanded

$$\frac{dy}{dx} = \left(\frac{dy}{db} \cdot \frac{db}{da} \right) \cdot \frac{da}{dx}$$

REVERSE MODE ⟶ isolate a gradient from the one before it

Let $(x_1, \dots, x_d)$ input values

$(x_{d+1}, \dots, x_D)$ intermediate variables

x_D output variable

03/10

OPTIMIZATION

Procedure to find max/min of function. It's important in machine learning because when we want to find the parameters during the training part, we have to maximize / minimize a suitable function in order to find the best value.

CONDITION FOR ∃ global/local MINIMUM

$f: \mathbb{R}^N \longrightarrow \mathbb{R}$ continuous and differentiable in $\mathbb{R}^N$ → if all patial derivatives exist and are continuous

WE DON'T CARE MAXIMUM

- LOCAL MINIMUM $x^* \in \mathbb{R}^N$ is a local minimum $\longleftrightarrow$ $f(x^*) \leq f(x)$ $\forall x: \|x - x^*\| < \varepsilon$
- GLOBAL MINIMUM $x^* \in \mathbb{R}^N$ is a global minimum $\longleftrightarrow$ $f(x^*) \leq f(x)$ $\forall x \in \mathbb{R}^N$
- STATIONARY POINT $x^* \in \mathbb{R}^N$ is a stationary point $\longleftrightarrow$ $\nabla f(x^*) = \emptyset$

WITHOUT THE EQUAL IN THE DEFINITION WE TALK ABOUT THE 'STRICT' VERSION

FIRST ORDER CONDITION

If x^* is a stationary point of local minimum for f, then $\nabla f(x^*) = \emptyset$ and x^* is a stationary point for f

x^* local minimum $\Rightarrow$ $\nabla f(x^*) = \emptyset$

SECOND ORDER CONDITION

f is twice differentiable $\longrightarrow$ f is differentiable and also ∇f

If $\nabla f(x^*) = \emptyset$ and $\nabla^2 f(x^*)$ (HESSIAN MATRIX) is positive definite, then x^* is a strict local minimum for f

x^* is a strict local minimum $\Leftarrow$ $\nabla f(x^*) = \emptyset$, $\nabla^2 f(x^*)$ pos. def.

ALGORITHMS

They compute only local minimum

1) ITERATIVE METHODS

$\vec{X}, \vec{X_1}, \vec{X_2}, \dots, \vec{X_K} \xrightarrow[K \to \infty]{} X^*$ — CONVERGES COMPONENT BY COMPONENT

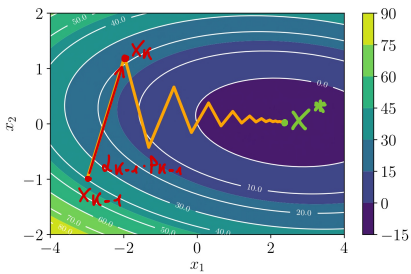
so $X_K = G(X_{K-1}, X_{K-2}) \longrightarrow X_K \simeq X^*$

DECENT METHOD $\longleftrightarrow$ $G(X_K) < G(X_{K-1})$

$\vec{X_K} = \vec{X_{K-1}} + \alpha_{K-1} \vec{P_{K-1}}$ — G ?

P_K : SEARCH DIRECTION

α_K : STEP LENGHT



CHOICE OF P

$\forall x$ . . .

If $p \in \mathbb{R}^N$, $p \neq \emptyset$ and $p^T \cdot \nabla f(x) < \emptyset$, then p is a decent direction for f in x

- **DESCENT DIRECTION**

$p \in \mathbb{R}^N$ is called descent direction for f in $x \longleftrightarrow \exists \alpha^* > 0 : f(\overbrace{x + \alpha p}^{X_{K+1}}) < f(\overbrace{x}^{X_K}) \quad \forall \alpha \in [0, \alpha^*]$

2) GRADIENT METHOD

$X_{K+1} = X_K - \alpha_K \nabla f(X_K)$

CHOICE OF α_K

α_K is constant ? $\longrightarrow$ NO $\longrightarrow$ the convergence is not guaranteed

BACKTRACKING PROCEDURE

$\alpha_K = 1$

IF $f(X_K - \alpha_K \nabla f(X_K)) > f(X_K) +$ NON SE LO RICORDA INFORMATIC NOTATION $\xrightarrow{?} F(w_t) - F(w_t + \alpha g_t) \geq M \alpha \left| g_t^T [\nabla F(w_t)] \right|$

Then α is not small enough, so $\alpha_K \leftarrow \alpha_K / c$

Then algorithm converges to a local minimum

HOW TO CHOOSE k^* WHERE TO STOP THE ALGORITHM

The gradient algorithm converge to a local minimum $\longrightarrow x_{k^*} \simeq x^*$

$\nabla f(x_{k^*}) \simeq 0 \longleftarrow \nabla f(x^*) = 0$

TOLERANCE $\tau \leftarrow \|\nabla f(x_k)\|_2 \simeq \tau$

RELATIVE: $\tau = \dfrac{\|\nabla f(x_{k^*})\|_2}{\|\nabla f(x_0)\|_2}$

ABSOLUTE: $\tau = \|\nabla f(x_{k^*})\|_2$

PSEUDOCODE OF GRADIENT METHOD

INPUT: f, x_0

```
k = 0
while [condition] :
    p_k = - ∇f(x_k)
    α_k backpropagating procedure
    x_(k+1) = x_k + α_k*p_k
    k += 1
return x_k
```

BACKTRACKING PROCEDURE

05/10

INITIALIZATION

In a sequence $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ we have $x_1 = x_0 - \alpha_0 \nabla f(x_0)$ at the beginning.

So x_0 is given in input and it's the seed to determine the whole sequence.

But how the start point can be chosen? ⟶ random(-1, 1) ⟶ may unfit the problem for different reasons

DIFFERENT x_0 LEAD TO DIFFERENT x^*

↓

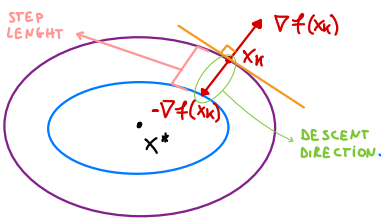
CHANGE THE INITIALIZATION POINT

ALGORITHM PROBLEMS

1) FLAT REGIONS AND LOCAL OPTIMA

It can lead to slowing down the algorithm or at worse it could even stop it.
But we can't know the shape before applying, so it occurs during execution.

2) DIFFERENTIAL CURVATURE



We draw the tangent to x_k in the level curves to a function.
The gradient would be perpendicular in that point and going out of the curves. Speed of descent depends on gradient magnitude.

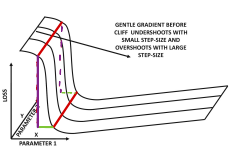
→ THEY ARE THE STEEPEST MOVEMENT BUT ONLY IN THAT POINT

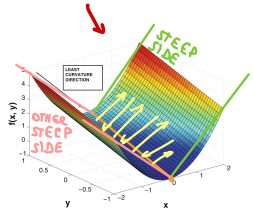
LEARNING RATE: $\|x_{k+1} - x_k\| = \alpha_k \|\nabla f(x_k)\|$ ⟶ step lenght

If step lenght is too high (if α_k is too large) the method could diverge

We may have: CLIFFS and VALLEYS

With a small step i obtain a much grater loss





The gradient-descent will bounce along the steep sides, without making much progress.

We also have the VANISCING AND EXPLODING GRADIENT PROBLEM

It occurs when we have large differences in the magnitudes of the partial derivatives with respect to parameters of different layers.

A solution to this problem is to standardize features before applying gradient descent.

3) NON-DIFFERENTIAL OBJECTIVE FUNCTIONS

Usually you know if a function it's differential or not, but if the function is implemented by a neural network we don't. NO DIFFERENTIAL ⟶ NO GRADIENT

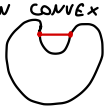
If only a few points are not differentiable we can slightly modify the computation of the gradient and going on.

If the are more non-differentiable points we have serious problems in the computation of the gradient

CONVEX FUNCTIONS

CONVEX SET

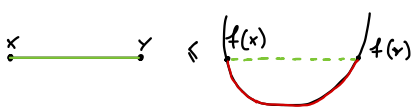
NON CONVEX SET



$f: \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}$

If Ω is a convex set and $\forall \vec{x}, \vec{y} \in \Omega$, $\forall \theta: 0 \leq \theta \leq 1$

Then $f(\theta x + (1-\theta) y) \leq \theta f(x) + (1-\theta) f(y)$ ⟶



QUADRATIC FUNCTION

SIMMETRIC AND DEFINITE POSITIVE

$f(\vec{x}) = \frac{1}{2} x^T A x + x^T \vec{b} + \vec{c}$

$\|Ax-b\|_2^2 = (Ax-b)^T (Ax-b) = x^T A^T A x - b^T A x - x^T A^T b - b^T b$ ⟹ so $\|Ax-b\|_2^2$ is quadratic function

PROPERTIES OF (STRICTLY) CONVEX FUNCTIONS

1) If f is convex
   Then any point of local minimum is also a global minimum ⟶ Δ there may be more global minimums
2) If f is strictly convex
   Then it exists a unique point of global minimum
3) If f is convex and differentiable
   Then any stationary point is a global minimum for f.

GRADIENT DESCENT WITH MOMENTUM

$$x_{i+1} = x_i - \alpha_i \nabla f(x_i) + \beta \Delta x_i$$

$$\Delta x_i = x_i - x_{i-1} \longrightarrow \text{SO WE MEMORIZE THE PREVIOUS 2 ITERATIONS}$$

STOCASTIC GRADIENT DESCENT (SGD)

When we work with $N \gg$ (IN NEURAL NETWORKS) we introduce much effort on gradient computation so we introduce an approximation in each step of gradient descent, called SGD.

STOCASTIC $\neq$ DETERMINISTIC

LOSS FUNCTION — NUM. DATA ELEMENTS

$$L(\theta) = \sum_{n=1}^{N} L_n(\theta) \longrightarrow \nabla L(\theta) = \sum_{n=1}^{N} \nabla L_n(\theta)$$

In practice the approximation is $\nabla L(\theta) \simeq \sum_{n \in S} \nabla L_n(\theta)$ $\quad S \subseteq \{1, \dots, N\}$ MINIBATCH

- When the learning rate decreases at an appropriate rate, stochastic gradient descent converges almost surely to local minimum.

EPOCH

Iterations necessary to use all the data. → 1 ITERATION ALL OVER THE DATA

ES:

EPOCH

$i = 1$ $\quad \nabla L(\theta) \simeq \nabla L_{n_1}(\theta)$ $\quad n_1 \in \{1, \dots, N\}$

$i = 2$ $\quad \nabla L(\theta) \simeq \nabla L_{n_2}(\theta)$ $\quad n_2 \in \{1, \dots, N\} - \{n_1\}$

$\vdots$

$i = p$ $\quad \nabla L(\theta) \simeq \nabla L_{n_p}(\theta)$ $\quad n_p \in \{1, \dots, N\} - \{n_1, \dots, n_{p-1}\}$

↳ 1 ELEMENT → EPOCH FINISHED

09/10

STATE SPACE (Ω)

It's the set of all the possible results of an experiment

ES: two coins → $\Omega = \{TT, HT, TH, HH\}$

EVENT (A)

It's a substate of Ω ($A \subset \Omega$) → Ω is particular event ($\Omega \subset \Omega$)

PROBABILITY (P)

It's a function $P(A) : S(A) \longrightarrow [0,1] \in \mathbb{R}$ $\forall A$ (S(A): set of all possible result)

$P(A) = \frac{\#(A)}{\#(\Omega)}$ (# CARDINALITY) → EQUALLY LIKELY MODEL

CONDITIONAL PROBABILITY

IF B, A are dependent, $P(B|A) = \frac{P(A \cap B)}{P(A)}$ where $P(A) \neq 0$

↳ $P(A \cap B) \neq P(A) \cdot P(B)$

↳ EXTENSIBLE TO $A_1, \dots, A_N$ EVENTS

RANDOM VARIABLES (X)

We need to map the events into mathematical objects.

$X : \Omega \longrightarrow \mathbb{R}$

ω ↑ $\quad X(\omega) = x$ ↓

TARGET SPACE: $T = \{x : x = X(\omega), \omega \in \mathbb{R}\}$ → SET OF ALL VALUES OF X

Random variable can be DISCRETE or CONTINUE

DISCRETE RV

T is constituted by a finite or countable numter of elements (∞ number with precedent and successive number)

CONTINUE RV

T is an interval of the union of intervals

DISCRETE RV CASE

PROBABILITY MASS FUNCTION (PMF) $\longrightarrow$ distribution

$f_X : T_X \to [0,1]$

$\forall x \in T_X, \quad f_X(x) = P(X = x)$

PROPERTIES

1) $f_X(x) \geq 0 \quad \forall x \in T_X$
2) $\sum_{x \in T_X} f_X(x) = 1$
3) $A \subset \Omega, \quad P(X = x \in A) = \sum_{x \in A} f_X(x)$

EXAMPLES

1) UNIFORM DISTRIBUTION : If $\#(T_X) = N$, $f_X(x) = 1/N$
2) POISSON DISTRIBUTION : If λ is the mean, $f_X(x) = e^{-\lambda} \cdot \frac{\lambda^x}{x!}$

CONTINUOUS RV CASE

PROBABILITY DENSITY FUNCTION (PDF)

$f_X : T_X \to \mathbb{R}$

$P(x \in A) = \int_A f_X(x)\, dx \xrightarrow{A=[a,b]} P(a \leq x \leq b) = \int_a^b f_X(x)\, dx$

! $P(a) = \int_a^a f_X(x)\, dx = 0$

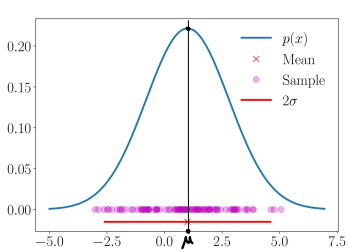
PROPERTIES

1) $f_X(x) \geq 0 \quad \forall x \in T_X$
2) $\int_{x \in T_X} f_X(x)\, dx = 1$
3) THE SAME OF DEFINITION

EXAMPLES

1) CONTINUOUS UNIFORM DISTRIBUTION : if $T_X = [a,b]$, then $f_X(x) = \frac{1}{b-a}$
2) NORMAL DISTIBUTION : $f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

↳ $(\mu, \sigma) = (0,1)$ we have standard normal distribution



10/10

STATISTICS

PROBABILITY ⟶ model of some process, where the underlying uncertainty is capture by random variables and we derive what happens

≠

STATISTICS ⟶ we observe that something has happened and we figure out the process that explains it

UNIVARIATE DISTRIBUTION

We consider only 1 random variable

BIVARIATE DISTRIBUTION

We consider only 2 random variable

MULTIVARIATE DISTRIBUTION

We consider more random variables

JOINT PROBABILITY

X, Y random variables

T_X, T_Y target spaces

$T_{XY} = T_X \times T_Y$

$$f_{XY} : T_{XY} \longrightarrow [0,1]$$

$$f_{XY}(x_i, y_i) = P(X = x_i, Y = y_i) = \frac{n_{XY}}{N}$$

MARGINAL PROBABILITY

Y \ X	x_1	x_2	. . .	x_k	$P_Y(y)$
Y_1	$P(x_1, y_1)$				$\sum_j^s P(x_i, y_j)$
⋮					⋮
Y_t				$P(x_k, y_t)$	
$P_X(x)$	$\sum_i^t P(x_i, y_j)$		. . .		1

Σ (down)

Σ (right)

$$P_X(x_i) = \sum_j P(x_i, y_j)$$ MARGINAL PROBABILITY OF X

$$P_Y(y_j) = \sum_i P(x_i, y_j)$$ MARGINAL PROBABILITY OF Y

! THERE IS ALSO FOR DENSITY FUNCTION !

CONDITIONAL PROBABILITY

$$P(Y = y_j \mid X = x_i) = \frac{n_{ij}}{c_i}$$ column for each x_i

$$P(X = x_i \mid Y = y_j) = \frac{n_{ij}}{r_j}$$ row for each y_j

SUM RULE

$$p(\boldsymbol{x}) = \begin{cases} \sum_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) & \text{if } \boldsymbol{y} \text{ is discrete} \\ \int_{\mathcal{Y}} p(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{y} & \text{if } \boldsymbol{y} \text{ is continuous} \end{cases}$$

PRODUCT RULE

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x}) = P(y,x)$$

BAYES THEOREM

how x and y are related

our subjective prior knowledge of x

$$\underbrace{p(\boldsymbol{x} \mid \boldsymbol{y})}_{\text{posterior}} = \frac{\overbrace{p(\boldsymbol{y} \mid \boldsymbol{x})}^{\text{likelihood}} \overbrace{p(\boldsymbol{x})}^{\text{prior}}}{\underbrace{p(\boldsymbol{y})}_{\text{evidence}}}$$

$$P(x,y) = P(y|x)\,P(x)$$
$$P(y,x) = P(x|y)\,P(y)$$
$$\Leftarrow P(y|x)\,P(x) = P(x|y)\,P(y)$$

our knowledge of y

We use it to know more about random variables to make inferences on unobservant RV.

QUANTITY OF INTEREST ⟶ encapsulates all available information from the prior and the data

↳ instead of considering the posterior in every computation, we consider only some statistic of the posterior (its maximum) ⟶ LOST OF INFORMATION ⟶ IF WE CONSIDER IT EVERY TIME IT WILL LEAD TO A QUICKER LEARNING

$$P(Y) = \int P(y|x)\,P(x)\,dx$$ MARGINAL LIKELIHOOD / EVIDENCE

STATISTIC

It's a deterministic function of that random variable.

SUMMARY STATISTIC ⟶ useful view of behaviour, summarize and characterize the distribution

EXPECTED VALUE

$$E_x[g(x)] = \begin{cases} \int_x g(x)P(x)\,dx & \text{continuous} \\ \sum_{x \in X} g(x)P(x) & \text{discrete} \end{cases}$$

$$\mathbb{E}_X[g(\boldsymbol{x})] = \begin{bmatrix} \mathbb{E}_{X_1}[g(x_1)] \\ \vdots \\ \mathbb{E}_{X_D}[g(x_D)] \end{bmatrix} \in \mathbb{R}^D$$ ⟶ MULTIVARIABLE

MEAN

$$\mathbb{E}_{X_d}[x_d] := \begin{cases} \int_{\mathcal{X}} x_d p(x_d) \mathrm{d}x_d & \text{continuous} \\ \sum_{x_i \in \mathcal{X}} x_i p(x_d = x_i) & \text{discrete} \end{cases}$$

$$\mathbb{E}_X[\boldsymbol{x}] = \begin{bmatrix} \mathbb{E}_{X_1}[x_1] \\ \vdots \\ \mathbb{E}_{X_D}[x_D] \end{bmatrix} \in \mathbb{R}^D$$ MULTIVARIABLE

COVARIANCE AND VARIANCE

$$\mathrm{Cov}[\boldsymbol{x}, \boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}\boldsymbol{y}^\top] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{y}]^\top = \mathrm{Cov}[\boldsymbol{y}, \boldsymbol{x}]^\top \in \mathbb{R}^{D \times E}$$

If you consider $y = x \longrightarrow \mathrm{Cov}(x,x) = \mathbb{E}[xx^T] - \mathbb{E}[x]\mathbb{E}[x]^T = V_x = \sigma^2$

CORRELATION

$$\mathrm{corr}[x, y] = \frac{\mathrm{Cov}[x, y]}{\sqrt{\mathbb{V}[x]\mathbb{V}[y]}} \in [-1, 1]$$

In machine learning, we search correlation to learn better.

EMPIRICAL MEAN

$$\bar{\boldsymbol{x}} := \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

EMPIRICAL COVARIANCE

$$\boldsymbol{\Sigma} := \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^\top = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^2$$

SUM AND TRANSFORMATION OF RANDOM VARIABLES

$$\mathbb{E}[x+y] = \mathbb{E}[x] + \mathbb{E}[y]$$
$$\mathbb{E}[x-y] = \mathbb{E}[x] - \mathbb{E}[y]$$
$$\mathbb{V}[x+y] = \mathbb{V}[x] + \mathbb{V}[y] + \mathrm{Cov}[x,y] + \mathrm{Cov}[y,x]$$
$$\mathbb{V}[x-y] = \mathbb{V}[x] + \mathbb{V}[y] - \mathrm{Cov}[x,y] - \mathrm{Cov}[y,x]$$

STATISTICAL INDEPENDENCE

X, Y are statistically independent if $P(x,y) = P(x)P(y)$

PROPERTIES

$p(\boldsymbol{y} \mid \boldsymbol{x}) = p(\boldsymbol{y})$
$p(\boldsymbol{x} \mid \boldsymbol{y}) = p(\boldsymbol{x})$
$\mathbb{V}_{X,Y}[\boldsymbol{x}+\boldsymbol{y}] = \mathbb{V}_X[\boldsymbol{x}] + \mathbb{V}_Y[\boldsymbol{y}]$
$\mathrm{Cov}_{X,Y}[\boldsymbol{x},\boldsymbol{y}] = \mathbf{0}$

CONDITIONAL INDEPENDENCE

X, Y are conditionally independent $\Leftrightarrow$ $P(x,y|z) = P(x|z)P(y|z)$

INNER PRODUCT OF RANDOM VARIABLES

$\langle X,Y \rangle = \mathrm{Cov}[x,y]$ but X, Y must have $\mu = 0$ (mean)

↳ SIMMETRIC AND POSITIVE DEFINITE

$\|x\| = \sqrt{\langle x,x \rangle} = \sqrt{\mathrm{Cov}[x,x]} = \sqrt{V[x]} = \sigma[x]$ LENGTH OF RV

$\cos\theta = \dfrac{\langle X,Y \rangle}{\|x\| \cdot \|y\|} = \dfrac{\mathrm{Cov}[x,y]}{\sqrt{V[x] \cdot V[y]}}$ ANGLE BETWEEN X, Y

$X \perp Y \leftrightarrow \langle X,Y \rangle = \varnothing = \mathrm{Cov}[x,y]$ ORTHOGONALITY

GAUSSIAN DISTRIBUTION

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

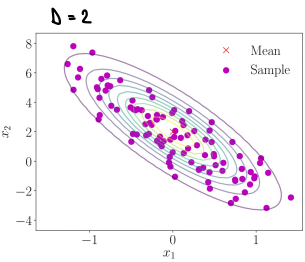
NOT A PROBABILISTIC DEPENDENCE BUT JUST A MATHEMATICAL ONE

MULTIVARIATE

DIMENSIONS

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

VARIANCE MATRIX



SUMS AND LINEAR TRANSFORMATION

If X,Y are independent gaussian variables then $p(\boldsymbol{x}+\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_x + \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_y)$

DATA, MODELS AND LEARNING

GOOD MODEL

Is a model which performs well on unseen data

MODELS

MODELS → FUNCTION (deterministic)

MODELS → PROBABILISTIC MODEL (probability distribution)

MODELS AS FUNCTIONS

We will consider only this class of functions: $f(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x} + \theta_0$

$\boldsymbol{\theta} = [\theta_0, \dots, \theta_D]^\top$

MODELS AS PROBABILITY DISTRIBUTION

We consider data as noisy and we want, by ML, to extract the information from the noise.

We will consider them as multivariate probability distribution.

LEARNING IS FINDING PARAMETERS

The right parameters will make my model a good model.

There are 3 algorithmic phases:

1) PREDICTION OR INFERENCE
2) TRAINING OR PARAMETER ESTIMATION
3) HYPERPARAMETER TUNING OR MODEL SELECTION

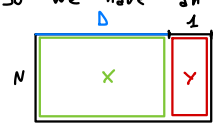
EMPIRICAL RISK MINIMIZATION

It's the learning phase for non-probabilistic model

HYPOTESIS CLASS OF FUNCTION

$x_n \in \mathbb{R}^D$ values and $y_n \in \mathbb{R}^n$ labels, by supervised learning setting we have $(x_1, y_1), \dots, (x_N, y_N)$

So we have an $N \times (D+1)$ matrix



And we hope to find a good parameter θ^* $\longrightarrow$ $f(x_n, \theta^*) \approx y_n$ ($= \hat{y}_n$)

LOSS FUNCTION FOR TRAINING

$\ell(y_n, \hat{y}_n)$

We assume $(x_1, y_1), \dots, (x_n, y_n)$ as independent and identically distributed

$X = [x_1, \dots, x_N]^T \in \mathbb{R}^{N \times D}$ and $y = [y_1, \dots, y_N]^T$

$$\mathbf{R}_{\text{emp}}(f, \boldsymbol{X}, y) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n)$$

IN LAB WE CONSIDERED ALL THIS AS LOSS FUNCTION

ES: $\ell(y_n, \hat{y}_n) = (y_n - \hat{y}_n)^2$ LEAST-SQUARES LOSS

we want $\min \{\ell(y_n, \hat{y}_n)\} = \theta^*$

$$\min_{\theta \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} (y_n - f(x_n, \theta))^2 \rightarrow \min_{\theta \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} (y_n - \theta^\top x_n)^2 \rightarrow \min_{\theta \in \mathbb{R}^D} \frac{1}{N} \|y - X\theta\|^2 = \theta^*$$

We want also to minimize the EXPECTED RISK $\mathbf{R}_{\text{true}}(f) = \mathbb{E}_{x,y}[\ell(y, f(x))]$ ($f(x) = \hat{y}_n$, $n = N+1, \dots, M$)

REGULARIZATION TO REDUCE OVERFITTING

Overfitting happens when the predictor fits too closely the training data and doesn't generalize well to new data. ↳ $R_{EMP}(f, X_{TRAIN}, Y_{TRAIN})$ understimates $R_{TRUE}(f)$

We find a compromise by using regularization by adding a REGULARIZATION PARAMETER and REGULATOR

ES: $\ell(y_n, \hat{y}_n) = (y_n - \hat{y}_n)^2$ LEAST-SQUARES LOSS

$$\theta^* = \min_{\theta} \frac{1}{N} \|y - X\theta\|^2 + \lambda \|\theta\|^2$$

PARAM, REGULATOR, $\lambda \cdot R(\theta)$, PENALTY TERM

16/10

PARAMETER ESTIMATION

We will use probability distribution to model our uncertainty

MAXIMUM LIKELIHOOD ESTIMATION (MLE)

- LIKELIHOOD → function related to probability distribution $\mathcal{L}_{\underline{x}}(\theta) = -\log\big(p(\underline{x} \mid \theta)\big)$ (FIXED)

IF we assume $(\underline{x}_1, \underline{y}_1), \dots, (\underline{x}_n, \underline{y}_n)$ independent and identically distributed so we have that

$$p(\underline{\mathcal{Y}} \mid \underline{\mathcal{X}}, \theta) = \prod_{n=1}^{N} p(y_n \mid x_n, \theta) \longrightarrow \mathcal{L}(\theta) = -\log\big(p(\mathcal{Y} \mid \mathcal{X}, \theta)\big) = -\sum_{n=1}^{N} \log\big(p(y_n \mid x_n, \theta)\big)$$

TO FIND GOOD θ WE MINIMIZE $L(\theta)$

ES: GAUSSIAN DISTRIBUTION

$y_n = x_n^T \theta$ for prediction and $\varepsilon_n \sim N(0, \sigma^2)$ (OBSERVATION UNCERTAINTY)

So we have that $p(y_n \mid x_n, \theta) = \mathcal{N}(y_n \mid x_n^\top \theta, \sigma^2)$ →

$$\begin{aligned}
\mathcal{L}(\theta) &= -\sum_{n=1}^{N} \log\big(p(y_n \mid x_n, \theta)\big) = -\sum_{n=1}^{N} \log\big(\mathcal{N}(y_n \mid x_n^\top \theta, \sigma^2)\big) \\
&= -\sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - x_n^\top \theta)^2}{2\sigma^2}\right)\right) \\
&= -\sum_{n=1}^{N} \log\left(\exp\left(-\frac{(y_n - x_n^\top \theta)^2}{2\sigma^2}\right)\right) - \sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) \\
&= \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - x_n^\top \theta)^2 - \sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right).
\end{aligned}$$

LEAST SQUARE (first term) — CONSTANT (second term)

WE CAN JUST MINIMIZE LEAST SQUARE

MINIMIZE $L(\theta)$ ←

$$\min_{\theta} \left\{ \sum_{n=1}^{N} (y_n - x_n^\top \theta)^2 \right\} \longleftarrow \min_{\theta} \left\{ \| Y - X\theta \|_2^2 \right\}$$

MAY SUFFER OF OVERFITTING

MAXIMUM A POSTERIORI ESTIMATION (MAP)

IF we have a prior knowledge x that makes us having more knowledge on θ.

$$p(\theta \mid x) = \frac{p(x \mid \theta) p(\theta)}{p(x)} \longrightarrow p(\theta, x) \propto p(x, \theta)\, p(\theta)$$

LINEAR REGRESSION

INDEPENDENT AND IDENTICALLY DISTRIBUTED RV THAT IDENTIFIES THE NOISE

We want to find a function f that maps $x \in \mathbb{R}^D$ in $f(x) \in \mathbb{R}$ with an error ε

$y_n = f(x_n) + \varepsilon$

In order to do that:

1) CHOICE OF MODEL AND PARAMETRIZATION

Given a dataset, which function classes can best model the data?

2) FINDING GOOD PARAMETERS

Given the model, which are the parameters that minimize the loss?

3) OVERFITTING AND MODEL SELECTION

Given too good parameters, how much error do I need to introduce to make the model more general

$y_n = f(x_n) + \varepsilon$ and $p(y|x) = \mathcal{N}(y | f(x), \theta)$ if $f = x^T\theta$ it's the previous case

We suppose that $f(x_n) = \underbrace{\phi^T(x)}_{\text{NONLINEAR TRANSFORMATION OF } x} \cdot \theta \longrightarrow y_n = f(x_n) + \varepsilon = \phi^T(x_n) \cdot \theta$ BECAUSE BEFORE IT WAS UNDERFITTING

$\phi^T(x)\theta = \sum_{n=1}^{N} \phi(x_n)\theta_n$ $\quad \phi : \mathbb{R}^D \to \mathbb{R}^K$ ($D > K$ USUALLY) and $\phi_i : \mathbb{R}^D \to \mathbb{R}$ ($\phi_i \leftrightarrow \psi_i$)

ES: POLYNOMIAL REGRESSION

We have $\phi_i(x) = x^i$ and so $f(x) = \sum_{i=0}^{k-1} \phi_i(x)\theta_i = \sum_{i=0}^{k-1} x^i \theta_i$

THE PREDICTIONS ARE NOT LINES ANYMORE BUT THEY ARE FUNCTIONS OF SETS OF POSSIBLE POLYNOMIAL

To solve it (like $f(x) = x^T\theta$) we have $L(\theta) = \frac{1}{2\sigma^2} \|y - \Phi\theta\|^2$ to be minimized so we use gradient $\nabla(\|y - \Phi\theta\|^2) = 0$

$\nabla(\|y - \Phi\theta\|^2) = \Phi^T\Phi\theta - \Phi^T y = 0 \longrightarrow \theta^* = \Phi^T y \cdot (\Phi^T\Phi)^{-1}$

17/10

POLYNOMIAL REGRESSION

1) DATA SET $(x_i, y_i) \longrightarrow y_i = G(x_i) + \varepsilon_i \qquad \varepsilon_i \sim N(0, \sigma^2)$

2) $\theta^* = \min_{\theta} \{ \| Y - \Phi\theta \|_2^2 \}$ by SGD

3) $f(x) = \sum_{i=0}^{K-1} \theta_i \cdot x^i$ polynomial

4) MAP $\rightarrow$ $\theta^*_{MAP} = \min_{\theta} \{ \| Y - \Phi\theta \|_2^2 + \lambda \| \theta \|_2^2 \}$ by SGD

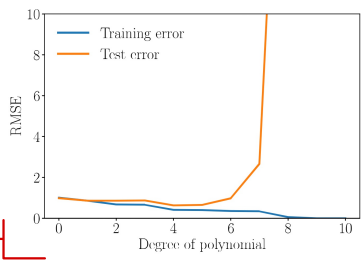
5) $f(x) = \sum_{i=0}^{K-1} \theta^*_{MAP} \cdot x^i$

RMSE (Root Mean Square Error)

RMSE $= \sqrt{\frac{1}{N} \|y - \Phi\theta\|^2} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (y_n - \phi^\top(x_n)\theta)^2}$

COMPARE ERRORS ON DATASET OF DIFFERENT DIMENSION

HAS SAME SCALE AND UNITS AS y_n



INCREASING THE EXPONENTIAL OF THE POLYNOMIAL WILL INTRODUCE MORE ERROR

NOT OVERFITTING ANYMORE